An Improved Semidefinite Programming Relaxation for the Satisfiability Problem

Miguel F. Anjos
Department of Electrical & Computer Engineering
University of Waterloo, Canada

Technical Report UW-E&CE#2002-09
13 June 2002

An Improved Semidefinite Programming Relaxation for the Satisfiability Problem

Miguel F. Anjos *
Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
anjos@stanfordalumni.org

13 June 2002

Abstract

The satisfiability (SAT) problem is a central problem in mathematical logic, computing theory, and artificial intelligence. An instance of SAT is specified by a set of boolean variables and a propositional formula in conjunctive normal form. Given such an instance, the SAT problem asks whether there is a truth assignment to the variables such that the formula is satisfied. It is well known that SAT is in general NP-complete, although several important special cases can be solved in polynomial time.

Semidefinite programming (SDP) refers to the class of optimization problems where a linear function of a matrix variable X is maximized (or minimized) subject to linear constraints on the elements of X and the additional constraint that X be positive semidefinite. We are interested in the application of SDP to satisfiability problems, and in particular in how SDP can be used to detect unsatisfiability.

In this paper we introduce a new SDP relaxation for the satisfiability problem. This SDP relaxation arises from the recently introduced paradigm of "higher liftings" for constructing semidefinite programming relaxations of discrete optimization problems. To derive the SDP relaxation, we first formulate SAT as an optimization problem involving matrices. Relaxing this formulation yields an SDP which significantly improves on the previous relaxations in the literature. The important characteristics of the SDP relaxation are its ability to prove that a given SAT formula is unsatisfiable independently of the lengths of the clauses in the formula, its potential to yield truth assignments satisfying the SAT instance if a feasible matrix of sufficiently low rank is computed, and the fact that it is more amenable to practical computation than previous SDPs arising from higher liftings. We present theoretical and computational results that support these claims.

*Research partially supported by a DO-NET Postdoctoral Research Fellowship and a Bell University Laboratories Research Grant. DO-NET is supported by the European Community within the Training and Mobility of Researchers Programme (contract number ERB TMRX-CT98-0202).

1 Introduction

The satisfiability (SAT) problem is a central problem in mathematical logic, computing theory, and artificial intelligence. An instance of SAT is specified by a set of boolean variables and a propositional formula in conjunctive normal form. Given such an instance, the SAT problem asks whether there is a truth assignment to the variables such that the formula is satisfied. It is well known that SAT is in general NP-complete, although several important special cases can be solved in polynomial time. There has been great interest in the design of efficient algorithms to solve the SAT problem; see [20] for an extensive survey.

Semidefinite programming (SDP) refers to the class of optimization problems where a linear function of a matrix variable X is maximized (or minimized) subject to linear constraints on the elements of X and the additional constraint that X be positive semidefinite. This includes linear programming problems as a special case, namely when all the matrices involved are diagonal. A variety of polynomial-time interior-point algorithms for solving SDPs have been proposed in the literature, and several excellent solvers for SDP are now available. We refer the reader to the recent handbook [39] for a thorough coverage of the theory and algorithms in this area, as well as several application areas where semidefinite programming researchers have made significant contributions.

We are interested in the application of SDP to satisfiability problems, and in particular in how SDP can be used to detect unsatisfiability. In [12, 11] de Klerk et al. introduced an SDP relaxation for SAT, the so-called Gap relaxation (defined in Section 1.2), which characterizes unsatisfiability for 2-SAT, i.e. when all the clauses have length at most two. (Note that 2-SAT is solvable in polynomial-time.) However, the Gap relaxation cannot detect unsatisfiability whenever all the clauses have length three or higher.

One of the main motivations for the improved SDP relaxation is to develop a semidefinite programming relaxation which can be used to prove that a given SAT formula is unsatisfiable, *independently of the lengths of the clauses* in the instance. Our SDP relaxation is easily defined for every instance of SAT, and it inherits all the favourable properties of the Gap relaxation proved in [12].

Our SDP relaxation is obtained using ideas from a "higher liftings" paradigm for constructing semidefinite programming relaxations of discrete optimization problems. The use of liftings was proposed in the literature within the framework of general purpose lift-and-project methods for 0-1 optimization (see for example [5, 34, 32]) and the idea of constructing semidefinite relaxations for 0-1 problems dates back to the introduction of the so-called theta function as a bound for the stability number of a graph [31]. To summarize the paradigm behind these higher semidefinite liftings, suppose that we have a discrete optimization problem on n binary variables. The SDP relaxation (or Shor relaxation) in the space of $(n+1)\times(n+1)$ symmetric matrices is called a first lifting. Note that the rows and columns of the matrix variable in this relaxation are indexed by the binary variables themselves. In this paradigm, we allow the semidefinite relaxations to have the rows and columns of the matrix variable indexed by *subsets* of the discrete variables in the formulation. These larger matrices can be interpreted as higher liftings, in the spirit of the second lifting proposed in [4], and its generalization independently proposed in [27, 28, 29].

This leads us to another motivation for this work, which is to address the question: Is

it possible to find "partial" liftings which are more amenable to practical computation than the entire higher liftings, while preserving their desirable properties? We believe that this work is a significant step towards a positive answer to this question. Indeed, we give two positive results in this direction, by considering both the theoretical and the computational point of view.

The theoretical results address both the satisfiability and unsatisfiability issues. It is straightforward to prove that if the SDP relaxation is infeasible, then the given SAT instance is unsatisfiable. On the other hand, if a feasible matrix Y is found, one interesting question is whether any information about the SAT instance might be obtained from the matrix Y. We provide sufficient conditions on the rank of a feasible matrix for it to yield a truth assignment that satisfies the SAT instance. These results are closely related to the recent work on SDP relaxations for Max-Cut in [1, 2] and [30], and further demonstrate the remarkable strength of the higher liftings.

On the computational side, we present results showing that in conjunction with an appropriate SDP solver, the new SDP relaxation was able to find either satisfying truth assignments, or certificates of unsatisfiability, for 3-SAT instances with up to 200 variables and 320 clauses. These dimensions represent a significant improvement over the computational results reported in [1], where second liftings for Max-Cut problems with only up to 27 binary variables were successfully solved. These results are also a step towards an SDP-based practical algorithm for satisfiability, which is the focus of ongoing research.

This paper is structured as follows. In the remainder of this section we introduce some notation, give a formal definition of the satisfiability problem, and discuss some previous work in the literature. In Section 2 we present the construction of the improved SDP relaxation and show how it relates to the previous SDP relaxation for SAT in the literature, namely the Gap relaxation. In Section 3 we state and prove the theoretical properties of the new SDP relaxation, and in Section 4 we present computational results that illustrate the strength of our approach. Finally, some directions for future research are outlined in Section 5.

1.1 Problem Definition and Notation

We consider the SAT problem in conjunctive normal form (CNF). An instance of SAT is specified by a set of variables $x_1, \ldots, x_n$ and a propositional formula $\Phi = \bigwedge_{j=1}^{m} C_j$, with each clause C_j having the form $C_j = \bigvee_{k \in I_j} x_k \vee \bigvee_{k \in J_j} \bar{x}_k$ where $I_j, J_j \subseteq \{1, \ldots, n\}$ and $\bar{x}_i$ denotes the negation of x_i. (We assume without loss of generality that $|I_j \cup J_j| \geq 2$ for every clause C_j.) The SAT problem is : Given a satisfiability instance, is Φ satisfiable, that is, is there a truth assignment to the variables $x_1, \ldots, x_n$ such that Φ evaluates to TRUE?

Special instances of SAT with certain constraints on the length of the clauses are often of particular interest, both theoretically and in practice. We use the notation k-SAT to refer to those instances of SAT for which each clause C_j satisfies $|I_j \cup J_j| \leq k$, and $\{k\}$-SAT for those instances where $|I_j \cup J_j| = k$ for every clause C_j.

For clause j and $k \in I_j \cup J_j$, define

$$s_{j,k} := \begin{cases} 1, & \text{if} \quad k \in I_j \\ -1, & \text{if} \quad k \in J_j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Using these variables, the SAT problem can be formulated as a $\{\pm 1\}$ integer programming problem. Let 1 denote TRUE and -1 denote FALSE, let $l(C_j) = |I_j \cup J_j|$ denote the number of literals in clause C_j, and represent each clause C_j by the corresponding inequality:

$$\sum_{k \in I_j \cup J_j} s_{j,k} x_k \geq 2 - l(C_j).$$

The SAT problem is now equivalent to the integer programming feasibility problem

$$\text{(IP-SAT)} \quad \begin{array}{ll} \text{find} & x \in \{\pm 1\}^n \\ \text{s.t.} & \displaystyle\sum_{k \in I_j \cup J_j} s_{j,k} x_k \geq 2 - l(C_j), \quad j = 1, \ldots, m \end{array}$$

Clearly the problem IP-SAT is as hard as the original SAT problem, and hence NP-complete. Some special cases of IP-SAT can be solved in polynomial time using linear programming, for example those studied in [9]. We refer to the survey [20] for more details.

1.2 The Gap Relaxation

de Klerk et al. [12, 11] introduced the Gap relaxation for SAT which is based on the elliptic approximations of clauses (see [37]) and is designed for detecting unsatisfiability. Indeed, they show that Gap characterizes unsatisfiability for 2-SAT problems. The characterization fails if any clause has length greater than two; in fact the Gap relaxation is always feasible whenever the instance has no clauses of length two. The Gap relaxation for 3-SAT may be expressed as follows:

$$\begin{array}{ll} \text{find} & X \in \mathcal{S}^{n+1} \\ \text{s.t.} & \\ & s_{j,i_1} s_{j,i_2} X_{i_1,i_2} - s_{j,i_1} X_{0,i_1} - s_{j,i_2} X_{0,i_2} + 1 = 0, \ \text{where } \{i_1, i_2\} = I_j \cup J_j, \ \text{if } l(C_j) = 2 \\ & s_{j,i_1} s_{j,i_2} X_{i_1,i_2} + s_{j,i_1} s_{j,i_3} X_{i_1,i_3} + s_{j,i_2} s_{j,i_3} X_{i_2,i_3} - s_{j,i_1} X_{0,i_1} - s_{j,i_2} X_{0,i_2} - s_{j,i_3} X_{0,i_3} \leq 0, \\ & \qquad \text{where } \{i_1, i_2, i_3\} = I_j \cup J_j, \ \text{if } l(C_j) = 3 \\ & \text{diag}\,(X) = e \\ & X \succeq 0 \end{array}$$

where $\mathcal{S}^n$ denotes the space of $n \times n$ symmetric matrices, $\text{diag}\,(X)$ represents a vector containing the diagonal elements of X, e denotes the vector of all ones, and $X \succeq 0$ denotes that X is positive semidefinite. The Gap relaxation is always feasible for instances of $\{3\}$-SAT, and hence is unable to detect unsatisfiability for such instances. Rounding schemes and approximation guarantees for the Gap relaxation, as well as its behaviour on so-called $(2+p)$-SAT problems, are studied in [11].

The Gap relaxation is based on the application of the elliptic approximations introduced in [37], and the linear constraints in the SDP are obtained by expanding and linearizing the elliptic approximation for each clause. Using these elliptic approximations, it is straightforward to extend the Gap relaxation to SAT instances where all the clauses have an arbitrary number of literals in each clause; however, the resulting relaxations are always feasible in the absence of clauses of length two, and hence unhelpful for detecting unsatisfiability. We will show that our SDP relaxation is able to detect unsatisfiability independently of the length of the SAT clauses, and furthermore that it inherits all the properties of the Gap relaxation.

2 Construction of the SDP Relaxation

We shall henceforth let TRUE be denoted by 1 and FALSE be denoted by -1. Our SDP relaxation is motivated by the following proposition.

Proposition 1 *For $l(C_j) \geq 2$, clause $C_j = \bigvee_{k \in I_j} x_k \vee \bigvee_{k \in J_j} \bar{x}_k$ is satisfied by $x_i = \pm 1, i \in I_j \cup J_j$, if and only if*

$$\sum_{t=1}^{l(C_j)} (-1)^{t-1} \left[\sum_{T \subseteq I_j \cup J_j, |T|=t} \left(\prod_{i \in T} s_{j,i} \right) \left(\prod_{i \in T} x_i \right) \right] = 1,$$

and it is not satisfied by $x_i = \pm 1, i \in I_j \cup J_j$, if and only if

$$\sum_{t=1}^{l(C_j)} (-1)^{t-1} \left[\sum_{T \subseteq I_j \cup J_j, |T|=t} \left(\prod_{i \in T} s_{j,i} \right) \left(\prod_{i \in T} x_i \right) \right] = 1 - 2^{l(C_j)}$$

where the coefficients $s_{j,i}$ are those defined in equation (1).

As special cases, we note that if $l(C_j) = 2$ then C_j is satisfied by $x_1, x_2 \in \{\pm 1\}$ if and only if

$$s_{j,1}x_1 + s_{j,2}x_2 - s_{j,1}s_{j,2}x_1x_2 = 1,$$

and if $l(C_j) = 3$ then C_j is satisfied by $x_1, x_2, x_3 \in \{\pm 1\}$ if and only if

$$s_{j,1}x_1 + s_{j,2}x_2 + s_{j,3}x_3 - s_{j,1}s_{j,2}x_1x_2 - s_{j,1}s_{j,3}x_1x_3 - s_{j,2}s_{j,3}x_2x_3 + s_{j,1}s_{j,2}s_{j,3}x_1x_2x_3 = 1.$$

Proof: By construction of the coefficients $s_{j,i}$, the clause is satisfied if and only if $s_{j,i}x_i$ equals 1 for at least one choice of i, or equivalently, if $\prod_{i \in I_j \cup J_j} (1 - s_{j,i}x_i) = 0$. Expanding the product, we have

$$\sum_{t=1}^{l(C_j)} (-1)^{t} \left[\sum_{T \subseteq I_j \cup J_j, |T|=t} \left(\prod_{i \in T} s_{j,i}x_i \right) \right] = -1.$$

The result follows. Similarly, the clause is not satisfied if and only if all the terms $s_{j,i}x_i$ equal -1, or equivalently, $\prod_{i \in I_j \cup J_j} (1 - s_{j,i}x_i) = 2^{l(C_j)}$. Again the result follows. ■

We note that the formulation of satisfiability using the polynomial constraint

$$\prod_{i \in I_j \cup J_j} (1 - s_{j,i} x_i) = 0$$

has been previously used in the design of SAT algorithms, see [19, 38].

Using Proposition 1, we now derive the SDP relaxation. First, let us formulate the satisfiability problem as follows:

$$\begin{array}{ll} \text{find} & x_1 \ldots, x_n \\ \text{s.t.} & \\ & \sum_{t=1}^{l(C_j)} (-1)^{t-1} \left[\sum_{T \subseteq I_j \cup J_j, |T|=t} \left(\prod_{i \in T} s_{j,i} \right) \left(\prod_{i \in T} x_i \right) \right] = 1, \quad j = 1, \ldots, m \\ & \\ & x_i^2 = 1, \quad i = 1, \ldots, n \end{array}$$

The next step is to formulate the problem in symmetric matrix space. Let P denote the set of nonempty sets $I \subseteq (I_j \cup J_j)$ such that the term $\prod_{i \in I} x_i$ appears in the above formulation. Also introduce new variables:

$$x_I := \prod_{i \in I} x_i,$$

for each $I \in P$, define the vector

$$v := (1, x_{I_1}, \ldots, x_{I_p})^T,$$

where p denotes the cardinality of P, and define the rank-one matrix

$$Y := vv^T,$$

whose rows and columns are indexed by $\emptyset \cup P$. By construction of Y, we have that $Y_{\emptyset,I} = x_I$ for all $I \in P$. Using these new variables, we can formulate the SAT problem as:

$$\begin{array}{ll} \text{find} & Y \in \mathcal{S}^{1+p} \\ \text{s.t.} & \\ & \sum_{t=1}^{l(C_j)} (-1)^{t-1} \left[\sum_{T \subseteq I_j \cup J_j, |T|=t} \left(\prod_{i \in T} s_{j,i} \right) Y_{\emptyset,T} \right] = 1, \quad j = 1, \ldots, m \\ & \\ & \operatorname{diag}(Y) = e \\ & \operatorname{rank}(Y) = 1 \\ & Y \succeq 0 \end{array}$$

Relaxing this formulation by omitting the rank constraint would give an SDP relaxation for SAT. However, in order to improve the SDP relaxation, we first add *redundant* constraints to this formulation. This approach of adding redundant constraints to the problem formulation so as to tighten the resulting SDP relaxation is discussed in detail for the Max-Cut problem in [3].

The constraint $\text{rank}\,(Y) = 1$ implies that for every triple I_1, I_2, I_3 of subsets of indices in P such that the symmetric difference of any two equals the third, the following three equations hold:

$$Y_{\emptyset,I_1} = Y_{I_2,I_3}, \quad Y_{\emptyset,I_2} = Y_{I_1,I_3}, \quad \text{and} \quad Y_{\emptyset,I_3} = Y_{I_1,I_2}. \tag{2}$$

Hence we can add some or all of these redundant constraints to the formulation (without affecting its validity). We add to the formulation above the equations of the form (2) for all the triples $\{I_1, I_2, I_3\} \subseteq P$ satisfying the symmetric difference condition. Beyond the fact that they tighten the SDP relaxation, the motivation for this particular choice of redundant constraints is that they are sufficient for proving the main theoretical result (Theorem 2).

Our final formulation of the SAT problem is thus:

$$\begin{array}{ll} \text{find} & Y \in \mathcal{S}^{1+p} \\ \text{s.t.} & \\ & \displaystyle\sum_{t=1}^{l(C_j)} (-1)^{t-1} \left[\sum_{T \subseteq I_j \cup J_j, |T|=t} \left(\prod_{i \in T} s_{j,i} \right) Y_{\emptyset,T} \right] = 1, \quad j = 1, \ldots, m \\ & \\ & Y_{\emptyset,I_1} = Y_{I_2,I_3}, \quad Y_{\emptyset,I_2} = Y_{I_1,I_3}, \quad \text{and} \quad Y_{\emptyset,I_3} = Y_{I_1,I_2}, \\ & \qquad \forall \{I_1, I_2, I_3\} \subseteq P : I_1 \Delta I_2 = I_3 \\ & \\ & \text{diag}\,(Y) = e \\ & \text{rank}\,(Y) = 1 \\ & Y \succeq 0 \end{array}$$

where $I_i \Delta I_i$ denotes the symmetric difference of I_i and I_j.

Remark 1 *If we had chosen P to contain all the subsets I with $|I| \leq K$, where K denotes the length of the longest clause in the SAT instance, and if we added all the redundant constraints of the form $Y_{I_1,I_2} = Y_{I_3,I_4}$, where $\{I_1, I_2, I_3, I_4\} \subseteq \emptyset \cup P$ and $I_1 \Delta I_2 = I_3 \Delta I_4$, then we would have obtained the Lasserre relaxation Q_{K-1} (as defined in [27]) for this problem. However, the resulting SDP has $p = O(n^K)$, which is far too large for practical computational purposes. Indeed, even for $K = 2$, the resulting relaxation cannot be solved in a reasonable time for $n \geq 30$, as demonstrated by the computational results in [1].*

The motivation for our approach is to construct an SDP relaxation which retains much of the tightness of Q_{K-1} while having a much smaller matrix variable as well as fewer linear constraints corresponding to symmetric differences. The matrix variable of our SDP relaxation has dimension $O(m * 2^K) = O(m)$, since for practical SAT instances K is a very small constant. The number of constraints is also $O(m)$: the SDP can have as many as $(\frac{1}{2}(2^K - 2)(2^K - 1) + 1)m$ linear constraints, but the presence of common variables between different clauses means that it will typically have many fewer constraints.

We illustrate the construction with the following example.

Example 1 *Suppose we are given the CNF formula*

$$(x_1 \vee x_2) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4 \vee x_5).$$

We construct the new SDP relaxation as follows. For the first clause, we have the variables x_1, x_2, and x_{12}. For the second clause, we add the variables x_3, x_4, x_{23}, x_{24}, x_{34}, and x_{234}. For the last clause, we add the variables x_5, x_{13}, x_{14}, x_{15}, x_{34}, x_{35}, x_{45}, x_{134}, x_{135}, x_{145}, x_{345}, and x_{1345}. The resulting matrix variable Y has dimension 21. As for the linear constraints, we add all the constraints which equate the elements of Y as depicted in Figure 1. (The elements of Y denoted by asterisks in Figure 1 are not involved in any of the linear equality constraints, although they are of course constrained by the positive semidefiniteness constraint.) Finally, for each clause in the CNF, we add one equality constraint:

$$\begin{aligned}
(x_1 \vee x_2) \Rightarrow \quad & Y_{\emptyset,x_1} + Y_{\emptyset,x_2} - Y_{\emptyset,x_{12}} = 1; \\
(x_2 \vee \bar{x}_3 \vee x_4) \Rightarrow \quad & Y_{\emptyset,x_2} - Y_{\emptyset,x_3} + Y_{\emptyset,x_4} + Y_{\emptyset,x_{23}} - Y_{\emptyset,x_{24}} + Y_{\emptyset,x_{34}} - Y_{\emptyset,x_{234}} = 1; \\
(\bar{x}_1 \vee x_3 \vee \bar{x}_4 \vee x_5) \Rightarrow \quad & -Y_{\emptyset,x_1} + Y_{\emptyset,x_3} - Y_{\emptyset,x_4} + Y_{\emptyset,x_5} + Y_{\emptyset,x_{13}} - Y_{\emptyset,x_{14}} + Y_{\emptyset,x_{15}} + Y_{\emptyset,x_{34}} \\
& -Y_{\emptyset,x_{35}} + Y_{\emptyset,x_{45}} + Y_{\emptyset,x_{134}} - Y_{\emptyset,x_{135}} + Y_{\emptyset,x_{145}} - Y_{\emptyset,x_{345}} - Y_{\emptyset,x_{1345}} = 1.
\end{aligned}$$

Finally, we write the resulting SDP relaxation as:

$$\begin{aligned}
\text{find} \quad & Y \in \mathbb{S}^{1+p} \\
\text{s.t.} \quad & \\
& Y_{\emptyset,x_1} + Y_{\emptyset,x_2} - Y_{\emptyset,x_{12}} = 1 \\
& Y_{\emptyset,x_2} - Y_{\emptyset,x_3} + Y_{\emptyset,x_4} + Y_{\emptyset,x_{23}} - Y_{\emptyset,x_{24}} + Y_{\emptyset,x_{34}} - Y_{\emptyset,x_{234}} = 1 \\
& -Y_{\emptyset,x_1} + Y_{\emptyset,x_3} - Y_{\emptyset,x_4} + Y_{\emptyset,x_5} + Y_{\emptyset,x_{13}} - Y_{\emptyset,x_{14}} + Y_{\emptyset,x_{15}} + Y_{\emptyset,x_{34}} \\
& \quad -Y_{\emptyset,x_{35}} + Y_{\emptyset,x_{45}} + Y_{\emptyset,x_{134}} - Y_{\emptyset,x_{135}} + Y_{\emptyset,x_{145}} - Y_{\emptyset,x_{345}} - Y_{\emptyset,x_{1345}} = 1 \\
& Y \text{ as in Figure 1} \\
& Y \succeq 0.
\end{aligned}$$

2.1 Connection to the Gap Relaxation

For any matrix Y feasible for the improved SDP relaxation, if we take the principal submatrix of Y corresponding to the rows and columns indexed by $\emptyset \cup \{x_i : i = 1, \ldots, n\}$, we obtain a positive semidefinite matrix with the same structure as the matrix variable in the Gap relaxation. It is clear that for clauses of length 2, the linear constraint expressing satisfiability is identical. For clauses of length 3, the linear constraint in the new SDP can be rewritten as

$$\begin{aligned}
& s_{j,i_1} Y_{0,i_1} + s_{j,i_2} Y_{0,i_2} + s_{j,i_3} X_{Y,i_3} - s_{j,i_1} s_{j,i_2} Y_{0,i_1 i_2} - s_{j,i_1} s_{j,i_3} Y_{0,i_1 i_3} - s_{j,i_2} s_{j,i_3} Y_{0,i_2 i_3} \\
& \qquad = 1 - s_{j,i_1} s_{j,i_2} s_{j,i_3} Y_{0,i_1 i_2 i_3},
\end{aligned}$$

where $\{i_1, i_2, i_3\} = I_j \cup J_j$. Now the positive semidefiniteness of Y implies that the right-hand side is always non-negative. If we relax the equation to this (implied) inequality, we get

$$s_{j,i_1} Y_{0,i_1} + s_{j,i_2} Y_{0,i_2} + s_{j,i_3} X_{Y,i_3} - s_{j,i_1} s_{j,i_2} Y_{0,i_1 i_2} - s_{j,i_1} s_{j,i_3} Y_{0,i_1 i_3} - s_{j,i_2} s_{j,i_3} Y_{0,i_2 i_3} \geq 0,$$

which is the constraint in the Gap relaxation. Hence, it follows that our SDP relaxation is always as least as tight as the Gap relaxation, and that the theoretical properties of the Gap relaxation also hold for our SDP relaxation.

$$Y = \begin{pmatrix}
1 & x_1 & x_2 & x_{12} & x_3 & x_4 & x_{23} & x_{24} & x_{34} & x_{234} & x_5 & x_{13} & x_{14} & x_{15} & x_{35} & x_{45} & x_{134} & x_{135} & x_{145} & x_{345} & x_{1345} \\
 & 1 & x_{12} & x_2 & * & * & * & * & * & * & x_{15} & x_3 & x_4 & x_5 & x_{135} & x_{145} & x_{34} & x_{35} & x_{45} & x_{1345} & x_{345} \\
 & & 1 & x_1 & x_{23} & x_{24} & x_3 & x_4 & x_{234} & x_{34} & * & * & * & * & * & * & * & * & * & * & * \\
 & & & 1 & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * & * \\
 & & & & 1 & x_{34} & x_2 & x_{234} & x_4 & x_{24} & x_{35} & x_1 & x_{134} & x_{135} & x_5 & x_{345} & x_{14} & x_{15} & x_{1345} & x_{45} & x_{145} \\
 & & & & & 1 & x_{234} & x_2 & x_3 & x_{23} & x_{45} & x_{134} & x_1 & x_{145} & x_{345} & x_5 & x_{13} & x_{1345} & x_{15} & x_{35} & x_{135} \\
 & & & & & & 1 & x_{34} & x_{24} & x_4 & * & * & * & * & * & * & * & * & * & * & * \\
 & & & & & & & 1 & x_{23} & x_3 & * & * & * & * & * & * & * & * & * & * & * \\
 & & & & & & & & 1 & x_2 & x_{345} & x_{14} & x_{13} & x_{1345} & x_{45} & x_{35} & x_1 & x_{145} & x_{135} & x_5 & x_{15} \\
 & & & & & & & & & 1 & * & * & * & * & * & * & * & * & * & * & * \\
 & & & & & & & & & & 1 & x_{135} & x_{145} & x_1 & x_3 & x_4 & x_{1345} & x_{13} & x_{14} & x_{34} & x_{134} \\
 & & & & & & & & & & & 1 & x_{34} & x_{35} & x_{15} & x_{1345} & x_4 & x_5 & x_{345} & x_{145} & x_{45} \\
 & & & & & & & & & & & & 1 & x_{45} & x_{1345} & x_{15} & x_3 & x_{345} & x_5 & x_{135} & x_{35} \\
 & & & & & & & & & & & & & 1 & x_{13} & x_{14} & x_{345} & x_3 & x_4 & x_{134} & x_{34} \\
 & & & & & & & & & & & & & & 1 & x_{34} & x_{145} & x_1 & x_{134} & x_4 & x_{14} \\
 & & & & & & & & & & & & & & & 1 & x_{135} & x_{134} & x_1 & x_3 & x_{13} \\
 & & & & & & & & & & & & & & & & 1 & x_{45} & x_{35} & x_{15} & x_5 \\
 & & & & & & & & & & & & & & & & & 1 & x_{34} & x_{14} & x_4 \\
 & & & & & & & & & & & & & & & & & & 1 & x_{13} & x_3 \\
 & & & & & & & & & & & & & & & & & & & 1 & x_1 \\
 & & & & & & & & & & & & & & & & & & & & 1
\end{pmatrix}$$

Figure 1: Structure of the (symmetric) matrix Y for Example 1

3 Properties of the SDP Relaxation

It is clear that if the propositional formula Φ is satisfiable, then using an assignment that makes the formula evaluate to TRUE it is straightforward to construct a rank-one matrix Y feasible for the SDP relaxation. The contrapositive of this statement gives a sufficient condition for proving unsatisfiability using the SDP relaxation.

Lemma 1 *Given a propositional formula in CNF, if the SDP relaxation is infeasible, then the CNF formula is unsatisfiable.*

Our next task is to derive sufficient conditions for proving satisfiability using the improved SDP relaxation. More specifically, we state and prove conditions on the rank of any matrix Y feasible for the SDP which guarantee that a truth assignment satisfying the SAT instance can be obtained from Y.

Our first condition follows immediately from [1, Theorem 2.1.1]:

Theorem 1 *If $Y \in \mathcal{S}^{1+p}$ and $Y \succeq 0$, then*

$$Y_{ij} = \pm 1 \text{ if and only if } Y \text{ is rank-one, i.e. } Y = vv^T \text{ for some } v \in \{\pm 1\}^{1+p}.$$

Hence, if Y is feasible for the SDP relaxation and all the entries of Y are equal to ± 1, then Y directly yields a truth assignment showing that the formula is satisfiable. In fact, it is easy to see that the above condition also applies to the Gap relaxation.

A much stronger rank condition holds for the improved SDP relaxation. We prove that if Y is feasible for the SDP relaxation and $\operatorname{rank} Y \leq 3$, then Y yields a truth assignment proving that the SAT instance is satisfiable. If we used the Lasserre relaxation Q_{K-1}, this result (and stronger ones) would follow from Theorem 21 of [30]. Interestingly, we can prove the above result for our significantly smaller relaxation. First we need the following proposition.

Proposition 2 *Suppose that* $\operatorname{rank} Y \leq 3$. *Then for every triple of subsets of indices I_1, I_2, I_3 such that the symmetric difference of any two equals the third, the four "triangle inequalities" hold for the corresponding entries in the first row of Y:*

$$x_{I_1} + x_{I_2} + x_{I_3} \geq -1,\ x_{I_1} - x_{I_2} - x_{I_3} \geq -1,\ -x_{I_1} - x_{I_2} + x_{I_3} \geq -1, \text{ and } -x_{I_1} + x_{I_2} - x_{I_3} \geq -1.$$

Furthermore, at least one of the four triangle inequalities holds with equality.

Proof: The rank assumption implies that the principal submatrix

$$Y = \begin{pmatrix} 1 & x_{I_1} & x_{I_2} & x_{I_3} \\ & 1 & x_{I_3} & x_{I_2} \\ & & 1 & x_{I_1} \\ & & & 1 \end{pmatrix}$$

has rank at most 3, and is positive semidefinite. The result follows from Theorem 4.4.7 of [1]. ■

Using this proposition, we can now prove that if clause C_j has three variables and Y_{C_j}, the 8×8 principal submatrix corresponding to it, has rank less than or equal to 3, then one of the off-diagonal elements equals ± 1.

Lemma 2 *Suppose that the matrix*

$$\begin{pmatrix} 1 & x_1 & x_2 & x_3 & x_{12} & x_{13} & x_{23} & x_{123} \\ & 1 & x_{12} & x_{13} & x_2 & x_3 & x_{123} & x_{23} \\ & & 1 & x_{23} & x_1 & x_{123} & x_3 & x_{13} \\ & & & 1 & x_{123} & x_1 & x_2 & x_{12} \\ & & & & 1 & x_{23} & x_{13} & x_3 \\ & & & & & 1 & x_{12} & x_2 \\ & & & & & & 1 & x_1 \\ & & & & & & & 1 \end{pmatrix}$$

is positive semidefinite and has rank less than or equal to 3. Then at least one of the off-diagonal elements equals ± 1.

Proof: For each triple of subsets of indices I_1, I_2, I_3 such that the symmetric difference of any two equals the third, we have the principal submatrix

$$\begin{pmatrix} 1 & x_{I_1} & x_{I_2} & x_{I_3} \\ & 1 & x_{I_3} & x_{I_2} \\ & & 1 & x_{I_1} \\ & & & 1 \end{pmatrix} \succeq 0,$$

and hence, by Lemma 2, one of the four triangle inequalities holds with equality. There are seven such triples of subsets; for each triple, we write the corresponding tight triangle inequality, obtaining the system:

$$\begin{aligned} \delta_1 x_1 + \delta_2 x_2 + \delta_3 x_{12} &= -1, \\ \delta_4 x_1 + \delta_5 x_3 + \delta_6 x_{13} &= -1, \\ \delta_7 x_2 + \delta_8 x_3 + \delta_9 x_{23} &= -1, \\ \delta_{10} x_{12} + \delta_{11} x_{13} + \delta_{12} x_{23} &= -1, \\ \delta_{13} x_1 + \delta_{14} x_{23} + \delta_{15} x_{123} &= -1, \\ \delta_{16} x_{12} + \delta_{17} x_3 + \delta_{18} x_{123} &= -1, \\ \delta_{19} x_{13} + \delta_{20} x_2 + \delta_{21} x_{123} &= -1, \end{aligned}$$

for appropriate choices of $\delta_i \in \{\pm 1\}, i = 1, \ldots, 21$.

Consider the first four equations, i.e. all the equations that do not involve x_{123}, and add them up, obtaining the equation:

$$(\delta_1 + \delta_4)x_1 + (\delta_2 + \delta_7)x_2 + (\delta_5 + \delta_8)x_3 + (\delta_3 + \delta_{10})x_{12} + (\delta_6 + \delta_{11})x_{13} + (\delta_9 + \delta_{12})x_{23} = -4. \quad (3)$$

All the parentheses are equal to either 0 or ± 2, and we have $4^4 = 256$ different possibilities, all of which fit into one of the following four cases:

All coefficients of Equation (3) equal 0 In this case, consider the addition of only the first three equations:

$$\delta_3 x_{12} + \delta_6 x_{13} + \delta_9 x_{23} = -3.$$

Now $Y \succeq 0$ and $\operatorname{diag} Y = e$ together imply that the absolute values of the entries of Y are bounded above by 1; thus the equation above implies that $x_{12} = -\delta_3$, $x_{13} = -\delta_6$, and $x_{23} = -\delta_9$ all equal ± 1.

Equation (3) has two nonzero coefficients Assume (without loss of generality) that the first two coefficients are nonzero. This implies that $|\frac{1}{2}(\delta_1 + \delta_4)| = |\frac{1}{2}(\delta_2 + \delta_7)| = 1$. Since equation (3) implies

$$\frac{1}{2}(\delta_1 + \delta_4)x_1 + \frac{1}{2}(\delta_2 + \delta_7)x_2 = -2$$

it follows (because $Y \succeq 0$ and $\operatorname{diag} Y = e$) that $x_1 = -\frac{1}{2}(\delta_1 + \delta_4)$ and $x_2 = -\frac{1}{2}(\delta_2 + \delta_7)$ both hold. Thus $|x_1| = |x_2| = 1$.

Equation (3) has four nonzero coefficients The structure of the system implies that among the four variables with nonzero coefficients, there exist three which are involved in the same equation in the system above. If we assume (without loss of generality) that the first four coefficients are nonzero, these variables are x_1, x_2, and x_{12}, for which $\delta_1 x_1 + \delta_2 x_2 + \delta_3 x_{12} = -1$. We also have that $\delta_1 = \delta_4$, $\delta_2 = \delta_7$, and $\delta_3 = \delta_{10}$ (otherwise their sums would equal zero), hence $\delta_4 x_1 + \delta_7 x_2 + \delta_{10} x_{12} = -1$. Adding these two equations, we have

$$(\delta_1 + \delta_4)x_1 + (\delta_2 + \delta_7)x_2 + (\delta_3 + \delta_{10})x_{12} = -2,$$

and thus $(\delta_5 + \delta_8)x_3 = -2$ holds. This implies $x_3 = -\frac{1}{2}(\delta_5 + \delta_8) = \pm 1$.

Equation (3) has all nonzero coefficients In this case, we prove that $x_{123} = \pm 1$. For convenience, first define

$$c_1 := \frac{1}{2}(\delta_1 + \delta_4), c_2 := \frac{1}{2}(\delta_2 + \delta_7), c_3 := \frac{1}{2}(\delta_5 + \delta_8),$$
$$c_{12} := \frac{1}{2}(\delta_3 + \delta_{10}), c_{13} := \frac{1}{2}(\delta_6 + \delta_{11}), c_{23} := \frac{1}{2}(\delta_9 + \delta_{12}),$$

all of which equal ± 1. It is straightforward to check that because of the structure of the system, only the following three possibilities can occur:

1. All c's equal 1: In this case, we have $x_1 + x_2 + x_3 + x_{12} + x_{13} + x_{23} = -2$. Now observe that by positive semidefiniteness,

$$e^T \begin{pmatrix} 1 & x_1 & x_2 & x_3 & x_{12} & x_{13} & x_{23} & x_{123} \\ & 1 & x_{12} & x_{13} & x_2 & x_3 & x_{123} & x_{23} \\ & & 1 & x_{23} & x_1 & x_{123} & x_3 & x_{13} \\ & & & 1 & x_{123} & x_1 & x_2 & x_{12} \\ & & & & 1 & x_{23} & x_{13} & x_3 \\ & & & & & 1 & x_{12} & x_2 \\ & & & & & & 1 & x_1 \\ & & & & & & & 1 \end{pmatrix} e \geq 0 \tag{4}$$

which immediately yields

$$1 + x_1 + x_2 + x_3 + x_{12} + x_{13} + x_{23} + x_{123} \geq 0.$$

Hence, $1 + x_{123} \geq 2$ holds; but again by positive semidefiniteness, $1 + x_{123} \leq 2$ holds, hence $x_{123} = 1$.

2. Three c's equal -1 and three equal 1: This can occur in only four different ways, namely that the coefficients equal to -1 are one of the four sets:

$$\{c_1, c_2, c_3\}, \{c_1, c_{12}, c_{13}\}, \{c_2, c_{12}, c_{23}\}, \{c_3, c_{13}, c_{23}\}.$$

We detail the proof for the first set, the other proofs are similar.
We have $-x_1 - x_2 - x_3 + x_{12} + x_{13} + x_{23} = -2$. Replacing e in (4) by the vector

$$\begin{pmatrix} 1 & c_1 & c_2 & c_3 & c_{12} & c_{13} & c_{23} & -1 \end{pmatrix}^T,$$

we obtain the inequality

$$1 - x_1 - x_2 - x_3 + x_{12} + x_{13} + x_{23} - x_{123} \geq 0.$$

Hence, $1 - x_{123} \geq 2$ holds; but again by positive semidefiniteness, $1 - x_{123} \leq 2$ holds, hence $x_{123} = -1$.

3. Four c's equal -1 and two equal 1: This last case can occur in only three different ways, namely that the coefficients equal to 1 are one of the three sets: $\{c_1, c_{23}\}, \{c_2, c_{13}\}, \{c_3, c_{12}\}$. Again we detail the proof for the first set, the others are proved similarly. We have $x_1 - x_2 - x_3 - x_{12} - x_{13} + x_{23} = -2$. Replacing e in (4) by the vector

$$\begin{pmatrix} 1 & c_1 & c_2 & c_3 & c_{12} & c_{13} & c_{23} & 1 \end{pmatrix}^T,$$

we obtain the inequality

$$1 + x_1 - x_2 - x_3 - x_{12} - x_{13} + x_{23} + x_{123} \geq 0.$$

Hence, $1 + x_{123} \geq 2$ holds; but $1 + x_{123} \leq 2$ holds, hence $x_{123} = 1$.

■

Supposing that $\operatorname{rank} Y \leq 3$, we now describe how to extract a truth assignment that satisfies all the clauses in the CNF formula. Let Y_{C_j} denote the principal submatrix corresponding to the rows and columns indexed by the $2^{l(C_j)}$ terms related to the variables in this clause (including $\emptyset$, i.e. the first row and column).

Lemma 3 *Suppose that Y is feasible for the SDP relaxation and* $\operatorname{rank} Y \leq 3$. *Then for each clause C_j, the corresponding principal submatrix Y_{C_j} can be expressed as a convex combination of (at most) three rank-one matrices representing truth assignments.*

Proof: There are three cases for the rank of Y_{C_j}. If $\operatorname{rank} Y_{C_j} = 1$ then by Theorem 1, all the entries are ± 1, so we are done.

If $\operatorname{rank} Y_{C_j} = 2$ then we can use the same approach as in Algorithm 5.2.3 in [1] and obtain two truth assignments that satisfy clause C_j.

If $\operatorname{rank} Y_{C_j} = 3$ then there are two cases. First, if $l(C_j) = 2$ then without loss of generality Y_{C_j} has the form

$$\begin{pmatrix} 1 & x_1 & x_2 & x_{12} \\ & 1 & x_{12} & x_2 \\ & & 1 & x_1 \\ & & & 1 \end{pmatrix} \succeq 0, \tag{5}$$

and one triangle inequality involving the three variables holds tight. Using this inequality, we can easily find the three corresponding triples of ± 1, and hence the three rank-one matrices and their convex combination representing (5).

If $l(C_j) = \lambda \geq 3$ then apply Lemma 2 to the principal submatrix of Y_{C_j} indexed by the power set of $\{\lambda - 2, \lambda - 1, \lambda\}$. By Lemma 2, one of the off-diagonal elements equals ± 1, say x_λ without loss of generality. Then each of the elements in Y_{C_j} whose indexing subscript contains λ can be expressed as $\pm$ some other element whose indexing subscript does not contain λ. Hence, we can reduce the problem to a principal submatrix of dimension $2^{l(C_j)-1}$. Repeating this argument until we are down to a 4×4 submatrix of the form (5), we can now obtain the three triples of ± 1 as above. Using the equations derived along the way, we obtain three λ-tuples of ± 1, and hence three rank-one matrices representing truth assignments. ■

Finally, whenever two clauses share one or more variables, they share a common principal submatrix. Thus, as we construct the various truth assignments satisfying each clause, we can ensure that they agree on the values assigned to these common variables. In this way, we obtain one (or more) truth assignment(s) for the entire CNF formula.

We summarize our theoretical results on the new relaxation as a theorem.

Theorem 2 *Given any propositional formula in CNF, consider the SDP relaxation constructed as presented in Section 2. Then*

- *If the SDP is infeasible, then the formula is unsatisfiable.*
- *If the SDP is feasible, and Y is a feasible matrix such that* $\operatorname{rank} Y \leq 3$*, then a truth assignment satisfying the formula can be obtained from Y. In the particular case that all the entries of Y equal ± 1, such a truth assignment is obtained from Y without any additional computation.*

4 Computational Results

We now present preliminary computational results that illustrate the potential of our SDP relaxation. We implemented the SDP relaxation for 3-SAT and experimented with the AIM set of problems from the DIMACS benchmark set of SAT problems available at `www.satlib.org`. This set has 72 instances of 3-SAT, 24 of which are unsatisfiable. We were able to solve the

SDPs for all the instances with at most 600 clauses (i.e. $m = 600$), which comprise 64 out of the 72 instances in the test set. The instances with 600 clauses took just under 16 hours of CPU time each.

Since we want to detect the possibility that the SDP relaxation is infeasible, two excellent choices of SDP solvers are

- SDPT3 [36] (available at `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`), and
- SeDuMi [35] (available at `http://fewcal.kub.nl/sturm/software/sedumi.html`).

Indeed, whenever infeasibility is detected, these solvers return a certificate of infeasibility, which, by Theorem 2, is for us a certificate of unsatisfiability. Also by Theorem 2, a feasible Y with $\operatorname{rank} Y \leq 3$ yields a certificate of satisfiability. Thus we can use the SDP relaxation to prove *either* satisfiability or unsatisfiability of the given SAT instance. Combining the SDPs with an enumerative branching procedure yields an algorithm for solving general instances of satisfiability.

All the SDPs in this paper have been stated as feasibility problems. In practice, we can choose an objective function to be optimized. We used the function $\sum_{i=1}^{n} x_i$ as objective function for all our tests. However, for the computational results reported here, we did not exploit any bounding information within the enumerative scheme.

We report results from applying our SDP-based approach to both satisfiable and unsatisfiable instances of SAT. For the computational results reported, we used the solver SDPT3 (version 3.0) with its default settings, accessed either via the optimization site NEOS (`http://www-neos.mcs.anl.gov/neos/`) [10, 13, 14, 18] or on a local implementation at the University of Waterloo.

4.1 Proving Unsatisfiability Using the SDP Relaxation

We explored the effectiveness of the SDP relaxation when applied to the unsatisfiable instances. First, for each of these instances, we set up and solved the SDP relaxation. Interestingly, the SDP immediately detected unsatisfiability of 2 instances out of 24. The dimensions and computational times for the SDPs are reported in Table 1.

We then explored the possibility of applying the SDP relaxation within an enumerative scheme by using it to prove unsatisfiability of the four `aim-50-1_6-no-*` instances and the four `aim-50-2_0-no-*` instances. Our branching strategy was depth-first search and the heuristic for choosing a variable x_i for branching very simple: if at the current node neither satisfiability nor unsatisfiability has been proved, branch on the variable x_i with smallest absolute value (with possible ties broken arbitrarily). The idea behind this branching rule is that for the rank-one matrices corresponding to truth assignments, all the entries equal ± 1; hence entries with small magnitude are not desirable. We found that, due to the tightness of the SDP relaxation, this simple strategy was quite effective. The enumerative scheme was implemented on a 300MHz SUNSparc at the University of Waterloo. The results are summarized in Table 2.

In the next section, we illustrate the application of the SDP to prove satisfiability. We found that it was also quite successful for this purpose.

Problem Number	Number of variables	Number of clauses	Number of constraints in the SDP	Size of Y	SDP is infeasible	Total CPU seconds
aim-50-1_6-no-1	50	80	1599	296	No	198
aim-50-1_6-no-2	50	80	1746	318	No	243
aim-50-1_6-no-3	50	80	1603	295	**Yes**	102
aim-50-1_6-no-4	50	80	1749	320	No	213
aim-50-2_0-no-1	50	100	2185	395	No	436
aim-50-2_0-no-2	50	100	2139	385	**Yes**	269
aim-50-2_0-no-3	50	100	2091	376	No	400
aim-50-2_0-no-4	50	100	2098	378	No	381
aim-100-1_6-no-1	100	160	3639	668	No	1785
aim-100-1_6-no-2	100	160	3466	639	No	1585
aim-100-1_6-no-3	100	160	3524	653	No	1602
aim-100-1_6-no-4	100	160	3599	662	No	1785
aim-100-2_0-no-1	100	200	4500	814	No	3234
aim-100-2_0-no-2	100	200	4422	801	No	3198
aim-100-2_0-no-3	100	200	4355	788	No	3046
aim-100-2_0-no-4	100	200	4366	791	No	2974
aim-200-1_6-no-1	200	320	7269	1344	No	13327
aim-200-1_6-no-2	200	320	7185	1343	No	13381
aim-200-1_6-no-3	200	320	7159	1315	No	13371
aim-200-1_6-no-4	200	320	7412	1366	No	13596
aim-200-2_0-no-1	200	400	8959	1626	No	25983
aim-200-2_0-no-2	200	400	8863	1612	No	26741
aim-200-2_0-no-3	200	400	8931	1620	No	25339
aim-200-2_0-no-4	200	400	8988	1634	No	26128

Table 1: Results for the unsatisfiable AIM Problems (computations performed via NEOS on a 400MHz Sparc2)

Problem Number	Number of SDPs solved to prove unsatisfiable (including root SDP)	Depth of branching tree	Total CPU seconds
aim-50-1_6-no-1	3	1	745
aim-50-1_6-no-2	15	4	5322
aim-50-1_6-no-3	1	0	197
aim-50-1_6-no-4	15	3	5091
aim-50-2_0-no-1	67	8	44593
aim-50-2_0-no-2	1	0	515
aim-50-2_0-no-3	21	6	11682
aim-50-2_0-no-4	3	1	1712

Table 2: Performance of the SDP in proving unsatisfiability for the smaller AIM Problems (computations performed on a 300MHz SUNSparc)

4.2 Proving Satisfiability Using the SDP Relaxation

We proceeded in a similar fashion to study the effectiveness of the SDP relaxation when applied to the satisfiable instances. For these instances, the SDP immediately found a truth assignment (in the form of a rank-1 matrix Y) for 11 out of 40 instances. The results are summarized in Tables 3, 4, and 5. We also tested the possibility of applying the SDP relaxation within an enumerative scheme to prove the satisfiability of the four `aim-50-1_6-yes1-*` instances and the four `aim-50-2_0-yes1-*` instances. The branching strategy was the same as above, and again we found that due to the tightness of the SDP relaxation, this simple strategy was quite effective, as demonstrated by the results in Table 6.

Problem Number	Number of clauses	Number of constraints in the SDP	Size of Y	SDP found a a satisfying assignment	Total CPU seconds
aim-50-1_6-yes1-1	80	1728	305	No	242
aim-50-1_6-yes1-2	80	1725	304	No	234
aim-50-1_6-yes1-3	80	1721	302	No	225
aim-50-1_6-yes1-4	80	1693	298	**Yes**	174
aim-50-2_0-yes1-1	100	2070	364	**Yes**	324
aim-50-2_0-yes1-2	100	2063	361	No	385
aim-50-2_0-yes1-3	100	2094	367	No	387
aim-50-2_0-yes1-4	100	2010	348	**Yes**	227
aim-50-3_4-yes1-1	170	3264	558	No	1259
aim-50-3_4-yes1-2	170	3327	566	No	1284
aim-50-3_4-yes1-3	170	3335	570	No	1378
aim-50-3_4-yes1-4	170	3217	548	No	1294
aim-50-6_0-yes1-1	300	5459	896	No	8163
aim-50-6_0-yes1-2	300	5301	872	No	5284
aim-50-6_0-yes1-3	300	5110	842	**Yes**	3868
aim-50-6_0-yes1-4	300	5249	864	**Yes**	4021

Table 3: Results for the satisfiable 50-variable AIM Problems (computations performed via NEOS on a 400MHz Sparc2)

Problem Number	Number of clauses	Number of constraints in the SDP	Size of Y	SDP found a a satisfying assignment	Total CPU seconds
aim-100-1_6-yes1-1	160	3464	610	**Yes**	1550
aim-100-1_6-yes1-2	160	3483	611	**Yes**	1413
aim-100-1_6-yes1-3	160	3527	624	**Yes**	1710
aim-100-1_6-yes1-4	160	3541	621	No	1588
aim-100-2_0-yes1-1	200	4252	743	No	2651
aim-100-2_0-yes1-2	200	4315	760	No	2888
aim-100-2_0-yes1-3	200	4215	736	**Yes**	2586
aim-100-2_0-yes1-4	200	4326	758	No	2689
aim-100-3_4-yes1-1	340	6777	1167	No	10304
aim-100-3_4-yes1-2	340	6752	1164	No	11791
aim-100-3_4-yes1-3	340	6660	1152	No	11059
aim-100-3_4-yes1-4	340	6748	1163	No	10080
aim-100-6_0-yes1-1	600	11158	1865	No	48860
aim-100-6_0-yes1-2	600	11170	1858	No	55711
aim-100-6_0-yes1-3	600	11317	1893	No	51304
aim-100-6_0-yes1-4	600	11266	1873	No	54803

Table 4: Results for the 100-variable satisfiable AIM Problems (computations performed via NEOS on a 400MHz Sparc2)

Problem Number	Number of clauses	Number of constraints in the SDP	Size of Y	SDP found a a satisfying assignment	Total CPU seconds
aim-200-1_6-yes1-1	320	7146	1255	No	14850
aim-200-1_6-yes1-2	320	7148	1255	**Yes**	15444
aim-200-1_6-yes1-3	320	7089	1248	**Yes**	14695
aim-200-1_6-yes1-4	320	7127	1254	No	12583
aim-200-2_0-yes1-1	400	8601	1515	No	28415
aim-200-2_0-yes1-2	400	8543	1508	No	26410
aim-200-2_0-yes1-3	400	8621	1513	No	22556
aim-200-2_0-yes1-4	400	8615	1515	No	22354

Table 5: Results for the satisfiable 200-variable AIM Problems (computations performed via NEOS on a 400MHz Sparc2)

Problem Number	Number of SDPs solved to find first satisfying truth assignment	Depth of first satisfying assignment found	Total CPU seconds to find first satisfying assignment
aim-50-1_6-yes1-1	4	2	1535
aim-50-1_6-yes1-2	10	6	3999
aim-50-1_6-yes1-3	4	2	1553
aim-50-1_6-yes1-4	1	0	451
aim-50-2_0-yes1-1	1	0	573
aim-50-2_0-yes1-2	22	6	14576
aim-50-2_0-yes1-3	18	6	11394
aim-50-2_0-yes1-4	1	0	425

Table 6: Performance of the SDP in proving satisfiability for the smaller AIM Problems (computations performed on a 300MHz SUNSparc)

4.3 Summary of the Computational Results

In summary, our computational results support the following conclusions:

1. The SDP relaxation can be solved in a few minutes of CPU time for instances with up to 50 variables and 100 clauses, and in several hours for instances with up to 200 variables and 600 clauses;

2. The SDP relaxation can effectively be used within a simple enumerative scheme to prove satisfiability or unsatisfiability of small SAT instances.

In comparison to other SAT algorithms in the literature, the computational effort required by our SDP-based approach is still too large for practical use. Nonetheless, the ability to compute with a higher semidefinite lifting for SAT problems with several hundreds of binary variables and clauses is a significant improvement on the results in [1] and [23].

5 Future Research

Our results raise several interesting questions for further research on our SDP relaxation.

From a theoretical point of view, one interesting question is to find classes of satisfiability problems besides those discussed in [12] for which it can be proved that the SDP relaxation always succeeds in detecting unsatisfiability, or always finds a satisfying truth assignment. Another interesting question is to study how this SDP could be applied to MAX-SAT problems, and how it might be useful in designing SDP-based approximation algorithms for MAX-SAT, an area which has been the focus of much recent research [24, 16, 17, 21].

From a computational point of view, we first observe that if one's main interest is to use the SDP relaxation to prove unsatisfiability, then one could test the feasibility of the SDP using the approach recently proposed in [6]. A second observation is that our SDP relaxation has a very particular structure, as exemplified in Figure 1. The question of how to solve SDPs by taking advantage of their intrinsic structure (or sparsity pattern) is currently an area of active research; see for example [7, 15, 33, 22, 25, 26, 8]. Beyond studying ways to exploit the structure of the SDP within a more specialized algorithm, we intend to introduce effective use of bounds within our enumerative scheme, and generally improve the efficiency of our algorithm. Further research work and computational experiments are ongoing.

Acknowledgements

The author thanks Hans Mittelman of Arizona State University for his technical support with the computational experiments performed through the NEOS server.

This research was partially done while the author was a DO-NET Postdoctoral Research Fellow at the Institut für Informatik of the University of Cologne. The author thanks Michael Jünger and his colleagues at the Institut für Informatik for their hospitality.

References

[1] M.F. Anjos. *New Convex Relaxations for the Maximum Cut and VLSI Layout Problems*. PhD thesis, University of Waterloo, 2001. Available online at `http://etd.uwaterloo.ca/etd/manjos2001.pdf`.

[2] M.F. Anjos and H. Wolkowicz. Geometry of semidefinite max-cut relaxations via matrix ranks. *J. Comb. Optim.*, 6(3):237–270, 2002.

[3] M.F. Anjos and H. Wolkowicz. Semidefinite programming for discrete optimization and matrix completion problems. *Discr. App. Math.*, 123/124:507–571, 2002.

[4] M.F. Anjos and H. Wolkowicz. Strengthened semidefinite relaxations via a second lifting for the max-cut problem. *Discr. App. Math.*, 119(1-2):79–106, 2002.

[5] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Programming*, 58(3, Ser. A):295–324, 1993.

[6] H.H. Bauschke and S.G. Kruk. The method of reflection-projection for convex feasibility problems with an obtuse cone. Technical report, Oakland University, Rochester MI, U.S.A., February 2002.

[7] S.J. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM J. Optim.*, 10(2):443–461 (electronic), 2000.

[8] S. Burer. Semidefinite programming in the space of partial positive semidefinite matrices. Technical report, Department of Management Sciences, University of Iowa, May 2002.

[9] M. Conforti and G. Cornuéjols. A class of logic problems solvable by linear programming. *J. Assoc. Comput. Mach.*, 42(5):1107–1113, 1995.

[10] J. Czyzyk, M. Mesnier, and J. Moré. The NEOS server. *IEEE J. on Computational Science and Engineering*, 5:68–75, 1998.

[11] E. de Klerk and H. Van Maaren. On semidefinite programming relaxations of $(2+p)$-SAT. *Annals of Mathematics of Artificial Intelligence*, to appear.

[12] E. de Klerk, H. Van Maaren, and J.P. Warners. Relaxations of the satisfiability problem using semidefinite programming. *J. Automat. Reason.*, 24(1-2):37–65, 2000.

[13] E. Dolan. The NEOS Server 4.0 administrative guide. Technical Report ANL/MCS-TM-250, Argonne National Laboratory, 2001.

[14] M. Ferris, M. Mesnier, and J. Moré. NEOS and Condor: Solving optimization problems over the Internet. *ACM Trans. on Math. Softw.*, 26(1):1–18, 2000.

[15] M. Fukuda, M. Kojima, K. Murota, and K. Nakata. Exploiting sparsity in semidefinite programming via matrix completion I: General framework. *SIAM J. Optim.*, 11(3):647–674 (electronic), 2000/01.

[16] M.X. Goemans and D.P. Williamson. New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7(4):656–666, 1994.

[17] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.

[18] W. Gropp and J. Moré. Optimization environments and the NEOS server. In M.D. Buhmann and A. Iserles, editors, *Approximation Theory and Optimization*, pages 167–182. Cambridge University Press, 1997.

[19] J. Gu. Global optimization for satisfiability (SAT) problem. *IEEE Transactions on Knowledge and Data Engineering*, 6(3):361–381, 1994.

[20] J. Gu, P.W. Purdom, J. Franco, and B.W. Wah. Algorithms for the satisfiability (SAT) problem: a survey. In *Satisfiability problem: theory and applications (Piscataway, NJ, 1996)*, pages 19–151. Amer. Math. Soc., Providence, RI, 1997.

[21] E. Halperin and U. Zwick. Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs. *J. Algorithms*, 40(2):184–211, 2001.

[22] C. Helmberg. A cutting plane algorithm for large scale semidefinite relaxations. Technical Report ZR-01-26, Konrad-Zuse-Zentrum Berlin, October 2001.

[23] D. Henrion and J.B. Lasserre. Gloptipoly - global optimization over polynomials with matlab and sedumi. Technical report, LAAS-CNRS, Toulouse, France, 2002.

[24] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proc. of 38th FOCS*, pages 406–415, 1997.

[25] M. Kočvara and M. Stingl. PENNON - a generalized augmented lagrangian method for semidefinite programming. Technical Report 286, Institute of Applied Mathematics, University of Erlangen, 2001.

[26] M. Kočvara and M. Stingl. PENNON - a code for convex nonlinear and semidefinite programming. Technical Report 290, Institute of Applied Mathematics, University of Erlangen, 2002.

[27] J.B. Lasserre. Optimality conditions and LMI relaxations for 0-1 programs. Technical report, LAAS-CNRS, Toulouse, France, 2000.

[28] J.B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM J. Optim.*, 11(3):796–817 (electronic), 2000/01.

[29] J.B. Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM J. Optim.*, 12(3):756–769 (electronic), 2002.

[30] M. Laurent. Semidefinite relaxations for max-cut. In M. Grötschel, editor, *The Sharpest Cut, Festschrift in Honor of M. Padberg's 60th Birthday.* SIAM, to appear.

[31] L. Lovász. On the Shannon capacity of a graph. *IEEE Trans. Inform. Theory*, 25(1):1–7, 1979.

[32] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim.*, 1(2):166–190, 1991.

[33] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima, and K. Murota. Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results. *Math. Prog.*, to appear.

[34] H.D. Sherali and W.P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math.*, 3(3):411–430, 1990.

[35] J.F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods Softw.*, 11/12(1-4):625–653, 1999.

[36] K.C. Toh, M.J. Todd, and R.H. Tütüncü. SDPT3—a MATLAB software package for semidefinite programming, version 1.3. *Optim. Methods Softw.*, 11/12(1-4):545–581, 1999.

[37] H. van Maaren. Elliptic approximations of propositional formulae. *Discrete Appl. Math.*, 96/97:223–244, 1999.

[38] J.P. Warners and H. van Maaren. A two-phase algorithm for solving a class of hard satisfiability problems. *Oper. Res. Lett.*, 23(3-5):81–88, 1998.

[39] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of semidefinite programming – Theory, algorithms, and applications*. Kluwer Academic Publishers, Boston, MA, 2000.